CS699
Lecture 4
Classification 1

Supervised vs. Unsupervised Learning

- Supervised learning (classification)
  - Supervision: The training data (observations, measurements, etc.) are accompanied by **class labels** indicating the class of the observations
  - New data is classified based on the training set
- Unsupervised learning (clustering)
  - The class labels of training data is unknown
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data

Prediction Problems: Classification vs. Numeric Prediction

- Classification
  - Predicts categorical class labels (discrete or nominal)
  - Constructs a model based on the training dataset which has known class labels and uses it to classify new data (or determine the class labels of new data)
- Numeric Prediction
  - Models continuous-valued functions, i.e., class attribute is a numeric attribute
  - Can be also used to predict missing values

Prediction Problems: Classification vs. Numeric Prediction

- Typical applications
  - Credit/loan approval:
  - Medical diagnosis: if a tumor is cancerous or benign
  - Fraud detection: if a transaction is fraudulent
  - Web page categorization: which category it is

Classification—A Two-Step Process

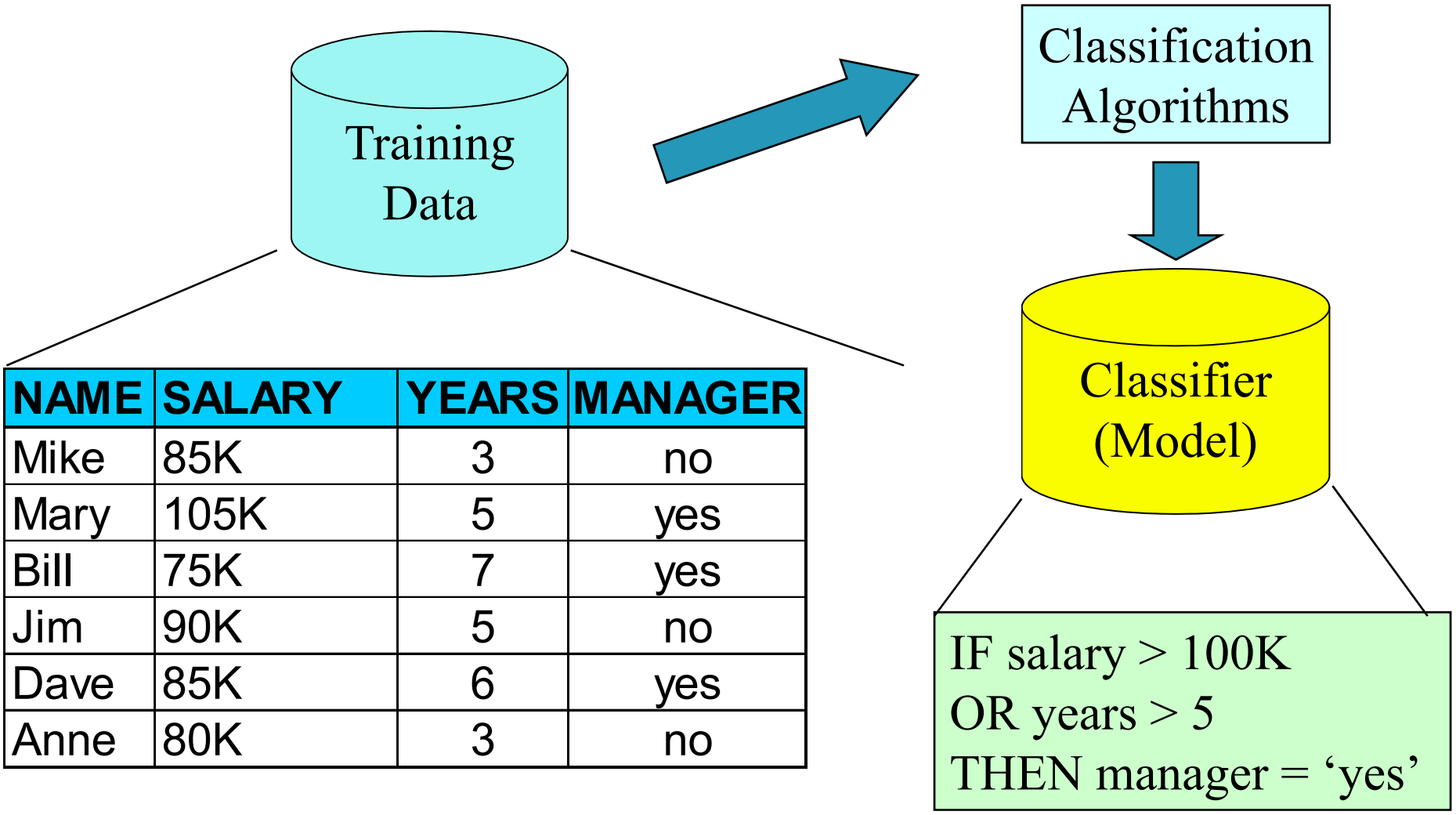
- Model construction: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction is training set
  - The model is represented as classification rules, decision trees, mathematical formulae, etc.

Classification—A Two-Step Process

- Model usage: Evaluate performance and use it to classify unseen tuples
  - Estimate performance of the model
    - The known class label of test tuple is compared with the classified result from the model
    - One performance measure, accuracy, is the percentage of test set tuples that are correctly classified by the model
    - Test set is independent of training set
  - If the performance is acceptable, use the model to classify data tuples with unknown class labels.

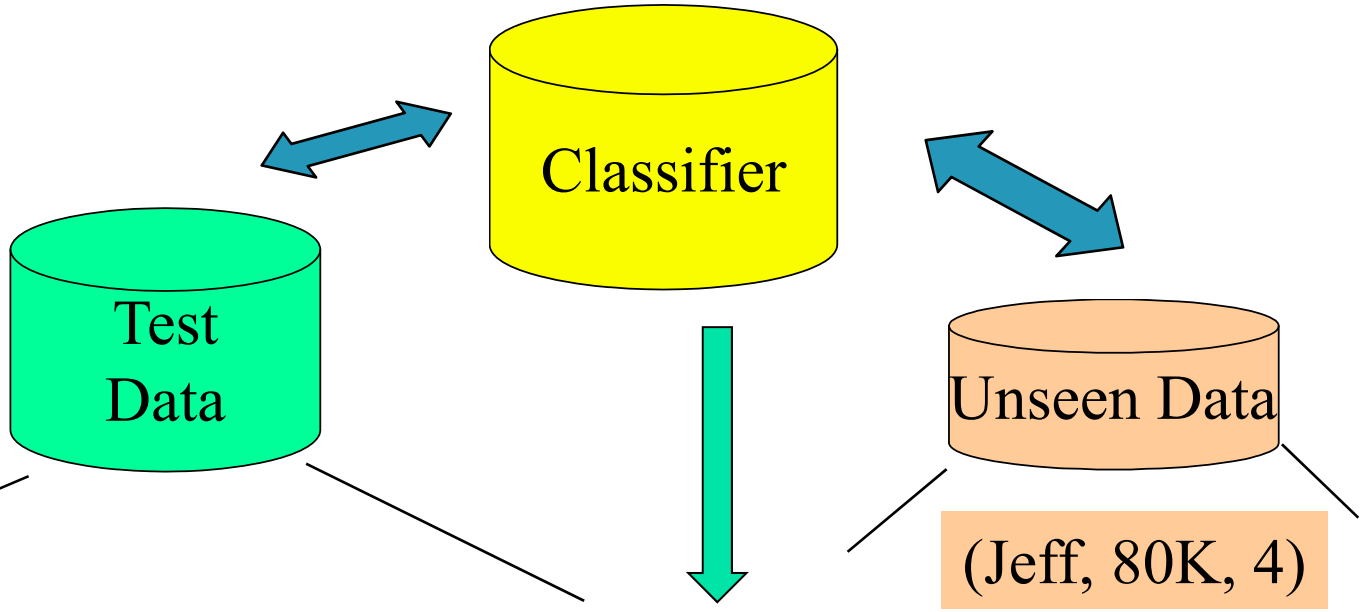
    Otherwise, build another model and repeat.

Process (1): Model Construction



NAME	SALARY	YEARS	MANAGER
Mike	85K	3	no
Mary	105K	5	yes
Bill	75K	7	yes
Jim	90K	5	no
Dave	85K	6	yes
Anne	80K	3	no

Process (2): Using the Model in Prediction



NAME	SALARY	YEARS	MANAGER	Test
Tom	90K	2	no	no
Merlisa	75K	4	yes	no
George	95K	6	yes	yes
Joseph	120K	5	yes	yes

Manager?
no

Accuracy = 75%

1R Classifier

- Simple
- Goal: Make rules based on a single attribute
- Compute the error rate for each attribute
- Choose the one with the smallest error rate

- Don't be confused by the name: Generate multiple rules, not a single rule.

1R Classifier

- Algorithm

For each attribute

For each value of the attribute

count the frequency of each class

find the most frequent class

make rule: assign that class to this attribute value

determine the error rate of this attribute value

Compute the total error rate of this attribute

Choose the rules of the attribute with the smallest error rate

1R Classifier

- Weather dataset

outlook	temperature	humidity	windy	play
sunny	hot	high	F	N
sunny	hot	high	T	N
overcast	hot	high	F	Y
rainy	mild	high	F	Y
rainy	cool	normal	F	Y
rainy	cool	normal	T	N
overcast	cool	normal	T	Y
sunny	mild	high	F	N
sunny	cool	normal	F	Y
rainy	mild	normal	F	Y
sunny	mild	normal	T	Y
overcast	mild	high	T	Y
overcast	hot	normal	F	Y
rainy	mild	high	T	N

1R Classifier

- Make rules for each attribute and calculate error rate

Attribute outlook

sunny: 2 yes's and 3 no's

rule: outlook = sunny → no, error rate = 2/5

overcast: 4 yes's and 0 no

rule: outlook = overcast → yes, error rate = 0/4

rainy: 3 yes's and 2 no's

rule: outlook = rainy → yes, error rate = 2/5

total error = 4/14

1R Classifier

- Make rules for each attribute and calculate error rate

Repeat the same for all other attributes.

Attribute temperature

temperature = hot → no (2/4), error rate = 2/4
(arbitrary tie breaking)
temperature = mild → yes (4/6), error rate = 2/6
temperature = cool → yes (3/4), error rate = 1/4

total error = 5/14

1R Classifier

Attribute humidity

humidity = high → no (4/7), error rate = 3/7

humidity = normal → yes (6/7), error rate = 1/7

total error = 4/14

Attribute windy

windy = false → yes (6/8), error rate = 2/8

windy = true → no (3/6), error rate = 3/6

(arbitrary tie breaking)

total error = 5/14

1R Classifier

- *outlook* and *humidity* have the same total error rate of 4/14. If we (randomly) choose *outlook*, the rules are

  If outlook = sunny → play = no

  If outlook = overcast → play = yes

  If outlook = rainy → play = yes

Example Dataset (RV Dataset)

Income	Marital	Age	Housing	Class
high	married	old	own	yes
high	married	young	own	yes
high	married	old	rent	yes
middle	married	young	own	yes
high	married	old	rent	yes
high	married	young	rent	yes
middle	married	old	rent	yes
high	single	young	own	no
high	single	old	own	yes
high	single	young	rent	no
low	married	old	own	yes
low	single	old	rent	yes
low	married	old	own	yes
low	single	young	own	no
middle	married	young	own	no
low	single	young	rent	no
low	married	young	rent	no
low	single	young	rent	no

Goal:

Build a classification model from this training dataset and predict using the model whether a customer would by an RV or not, assuming we know their income, marital status, age, and housing.

Bayesian Classification

- A statistical classifier: performs *probabilistic prediction, i.e.,* predicts class membership probabilities
- Foundation: Based on Bayes' Theorem.
- Performance: A simple Bayesian classifier, *naïve Bayesian classifier*, has comparable performance with decision tree and selected neural network classifiers
- Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct — prior knowledge can be combined with observed data
- Standard: Even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured

Bayesian Theorem

- Based on Bayes' rule (or Bayes' theorem) of conditional probability:

$$P(H \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid H)P(H)}{P(\mathbf{X})}$$

- P(H) (*prior*): *prior probability* of some *hypothesis* H
- P(X) (*evidence*)*:* probability that a sample X is observed
- P(X|H) (*likelihood*): the probability of observing the sample X, given that the hypothesis H holds
- P(H|X) (*posteriori*): the probability the hypothesis H holds given X

- posteriori = (likelihood x prior)/evidence

Bayesian Theorem

- A simple example of using Bayes' theorem for inference: If a person has muscle pain, what is the probability that the person has flu?

Bayesian Theorem

- F: A patent has flu, M: A patient has muscle pain
- We know from historical data;

  P(M|F) = 0.75 (if a person catches flu, he/she has muscle pain 75% of the time)

  P(F) = 0.00002 (the probability that a person has flu)

  P(M) = 0.005 (the probability that a person has muscle pain)
- The probability that a person has flu if that person has muscle pain:

$$P(F|M) = \frac{P(M|F)P(F)}{P(M)}$$ = 0.75 * 0.00002 / 0.005 = 0.003

Bayes' Theorem for Classification

- In classification, a hypothesis is "a sample belongs to a class." For example, a customer will (or will not) buy an RV.
- Rewrite Bayes theorem by replacing *H* with C_i, where C_i is a class

$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$

- In the example dataset,
  - there are two classes:
    - C_1: class = yes
    - C_2: class = no
  - *X* is a customer

Bayes' Theorem for Classification

- For the *RV* dataset:

$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}, \quad i = 1,2$$

- $P(C_i|X)$: Given attribute values of a customer X, the probability X will (or will not) buy an RV – this is classification
- $P(C_i)$: A customer will (or will not) buy an RV, regardless of age, income – this is class prior probability
- P(X)*:* When a customer is randomly picked, the probability that the customer's attribute values are the same as those of X
- $P(X|C_i)$: Given a customer buys (or does not buy) an RV, the probability that the customer's attribute values are the same as those of X

Bayes' Theorem for Classification

- Classification: Calculate and compare the following two probabilities and assign the class with the higher probability to X (or we predict the class label of X is the class with the higher probability).

$$P(class = yes \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid class = yes)P(class = yes)}{P(\mathbf{X})}$$

$$P(class = no \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid class = no)P(class = no)}{P(\mathbf{X})}$$

Bayes' Theorem for Classification

- In the equations, P(**X**) is the common for all classes. So we compute and compare only the numerators.

$$P(\mathbf{X}|C_i)P(C_i)$$

- For class label with two values, *yes* and *no*, we compute and compare the following two:

$$P(\mathbf{X}|class = yes)P(class = yes)$$

$$P(\mathbf{X}|class = no)P(class = no)$$

Derivation of Naïve Bayes Classifier

- We need to compute $P(X|C_i)$ and $P(C_i)$.
- $P(C_i)$ can be easily estimated from the training dataset

- For the RV dataset, there are two classes, yes and no. Let yes be C_1 and no be C_2.
- We can estimate $P(C_i)$ as follows:
  - $P(C_1)$ = (# yes tuples) / (total # tuples)
  - $P(C_2)$ = (# no tuples) / (total # tuples)

Derivation of Naïve Bayes Classifier

- Computation of $P(X|C_i)$ is not easy.
- It can be simplified with *class-conditional independence assumption* (a naïve assumption)

- *class-conditional independence* assumption: non-class attributes are conditionally independent (i.e., no dependence relation among non-class attributes).

- This greatly reduces the computation cost

Derivation of Naïve Bayes Classifier

- Based on this assumption, we compute $P(x_k|C_i)$ for each attribute x_k, and multiply them all to obtain $P(X|C_i)$ (this is possible because we assumed all attributes are independent of each other). Here, x_k is the value of attribute A_k for tuple X.

- $P(x_k|C_i)$ is the # of tuples in C_i having value x_k for A_k divided by $|C_{i,D}|$ (# of tuples of C_i in D)

$$P(\mathbf{X}|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i) \times P(x_2|C_i) \times \ldots \times P(x_n|C_i)$$

Naïve Bayesian Classifier: Training Dataset

Example: Classify a customer

X = (income = middle, marital = single, age = young, housing = own)

A summary table helps:

		yes	no	Total
Income	high	6	2	8
	low	3	4	7
	middle	2	1	3
	Total	11	7	18
Marital	married	9	2	11
	single	2	5	7
	Total	11	7	18
Age	old	8	0	8
	young	3	7	10
	Total	11	7	18
Housing	own	6	3	9
	rent	5	4	9
	Total	11	7	18

Naïve Bayesian Classifier: An Example

- Class prior probabilities are:

  $P(C_1)$ = P(yes) = 11/18 = **0.611**

  $P(C_2)$ = P(no) = 7/18= **0.389**

- Next, we compute $P(X|C_i)$ for each class
- For yes class:

  P(income = middle | class = yes) = 2/11 = 0.182

  (among 11 yes tuples, 2 have middle income)

  P(marital = single | class = yes) = 2/11 = 0.182

  (among 11 yes tuples, 2 are single)

  P(age = young | class = yes) = 3/11 = 0.273

  (among 11 yes tuples, 3 are young)

  P(housing = own | class = yes) = 6/11 = 0.545

  (among 11 yes tuples, 6 own house)

  P(X|class = yes) = 0.182 * 0.182 * 0.273 * 0.545 = 0.0049

Naïve Bayesian Classifier: An Example

- For no class

  P(income = middle | class = yes) = 1/7 = 0.143

  (among 7 no tuples, 1 has middle income)

  P(marital = single | class = yes) = 5/7 = 0.714

  (among 7 no tuples, 5 are single)

  P(age = young | class = yes) = 7/7 = 1.0

  (among 7 no tuples, All 7 are young)

  P(housing = own | class = yes) = 3/7 = 0.429

  (among 7 no tuples, 3 own house)

  P(X|class = no) = 0.143 * 0.714 * 1.0 * 0.429 = 0.0438

Naïve Bayesian Classifier: An Example

$P(X|C_i)*P(C_i)$:

$P(X|\text{class} = \text{yes}) * P(\text{class} = \text{yes}) = 0.0049 * 0.611 = 0.0030$

$P(X|\text{class} = \text{no}) * P(\text{class} = \text{no}) = 0.0438 * 0.389 = 0.0170$

The model predicts that X belongs to class no (or X will not buy an RV)

Naïve Bayesian Classifier: Comments

- Advantages
  - Easy to implement
  - Good results obtained in most of the cases
- Disadvantages
  - Assumption: class conditional independence, therefore loss of accuracy
  - Practically, dependencies exist among variables:
    - patient_profile: age, height, weight, etc.
    - Dependencies among these cannot be modeled by Naïve Bayesian Classifier
- How to deal with these dependencies? Bayesian Belief Networks (Chapter 9)

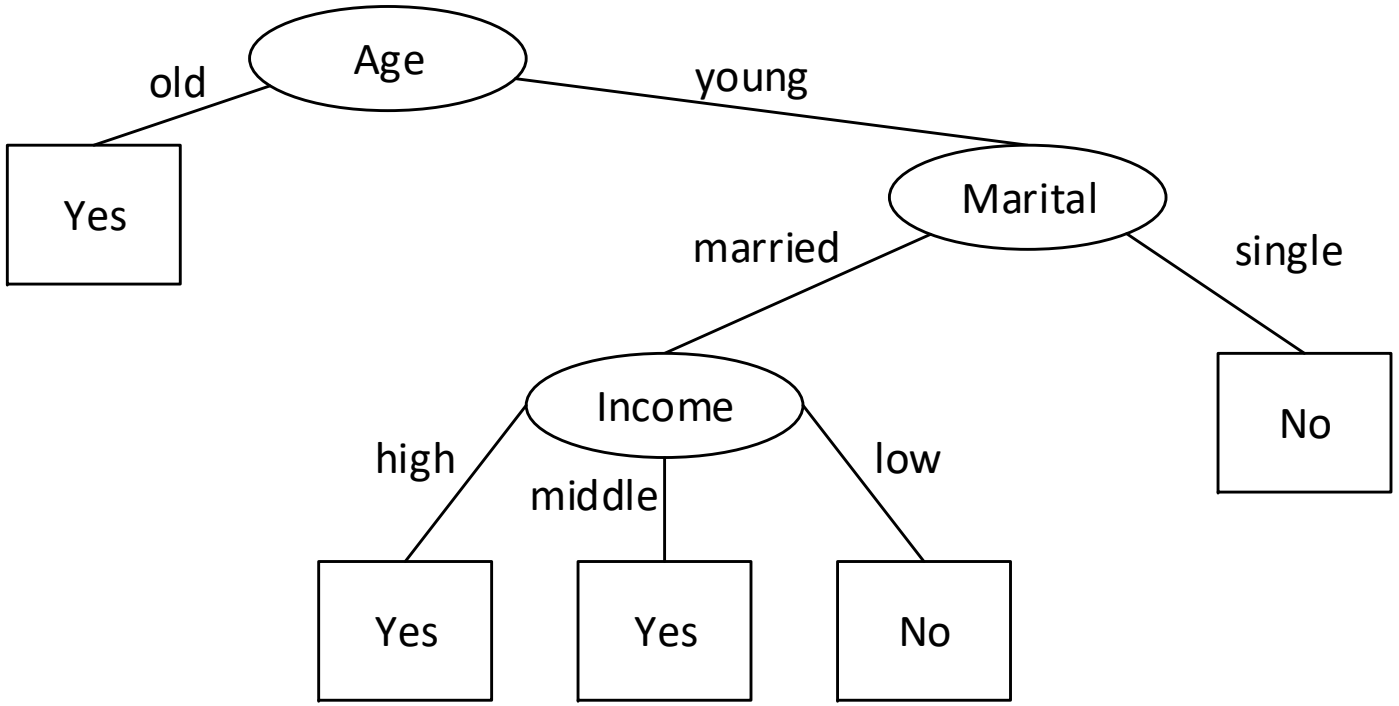
Decision Tree

- A popular classifier
- A model is built as a decision tree

- Internal node represents a test on an attribute
- Branch represents outcome of test
- Leaf node has a class label

Decision Tree Induction Example

- Training data set: RV dataset
- Decision tree model generated by the Ross Quinlan's ID 3 decision tree algorithm



Algorithm for Decision Tree Induction

- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a top-down, recursive manner
  - Attributes are categorical (if continuous-valued, they are discretized first)
  - At start, all training samples are considered to be at the root.

Algorithm for Decision Tree Induction

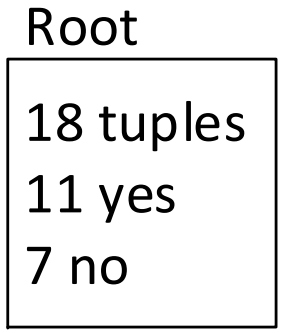
- Basic algorithm (continued)
  - A test attribute is selected on the basis of a heuristic or statistical measure (e.g., information gain)
  - samples are partitioned based on the selected attribute – children nodes are created
  - The same process is repeated at each child node (recursively)

Algorithm for Decision Tree Induction

- Conditions for stopping partitioning
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left
  - Minimum number of tuples in a node

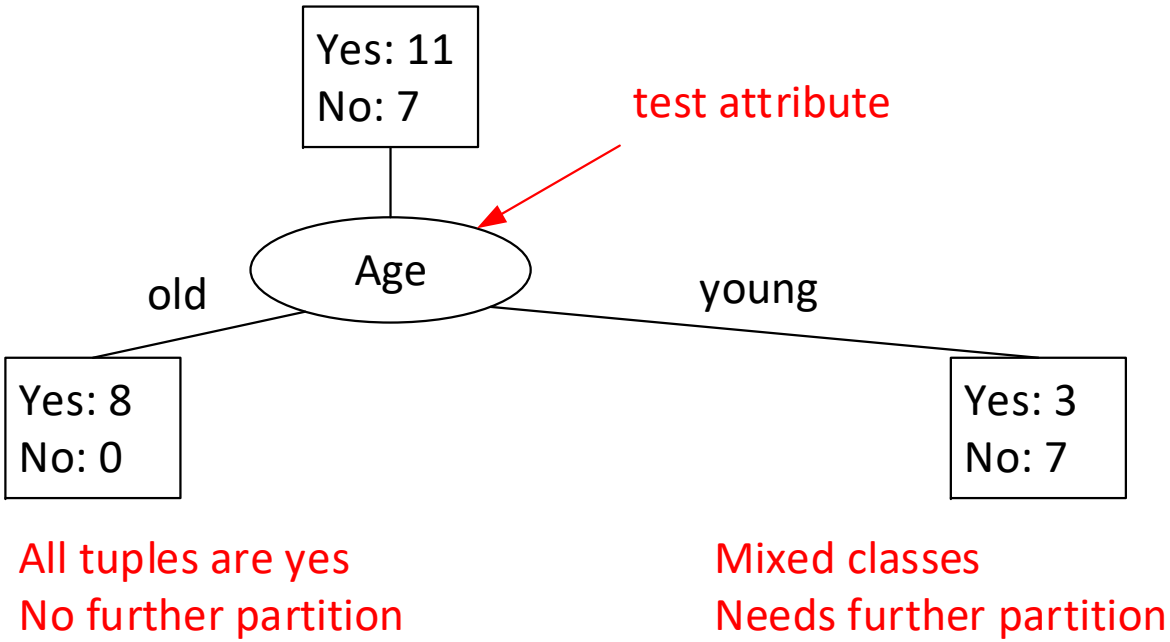
Algorithm for Decision Tree Induction

- Initially all samples are associated with the root node.



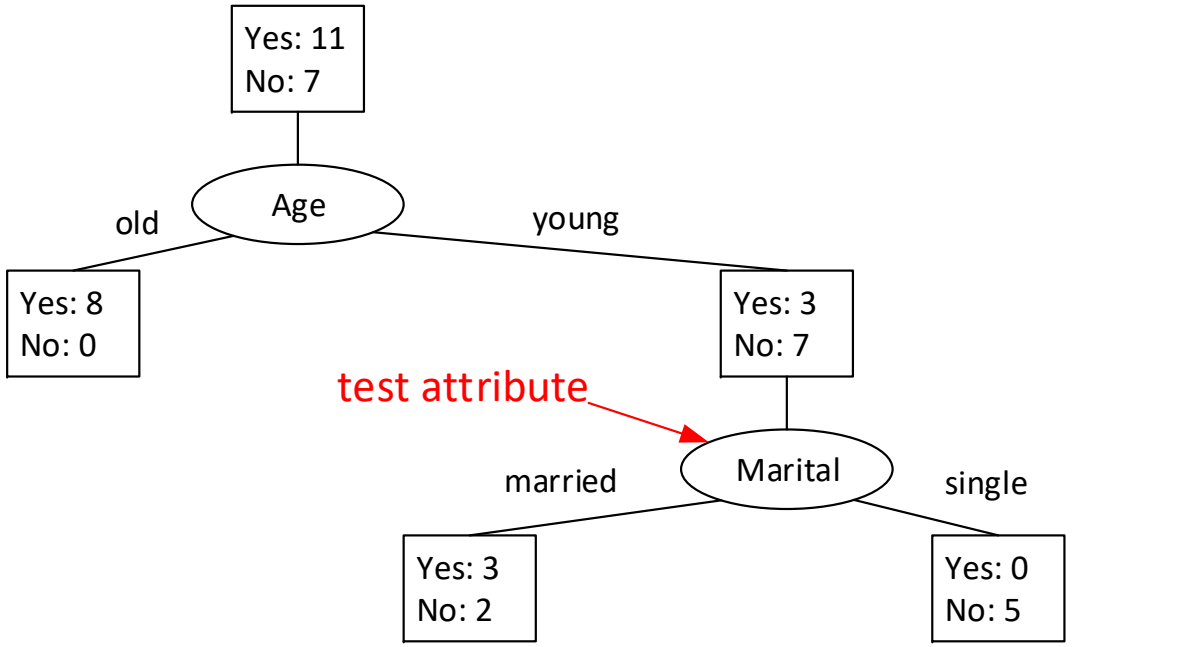
Algorithm for Decision Tree Induction

- *Age* is chosen as the test attribute (how it is chosen will be discussed later), and samples are partitioned into two nodes based on the value of *Age*.



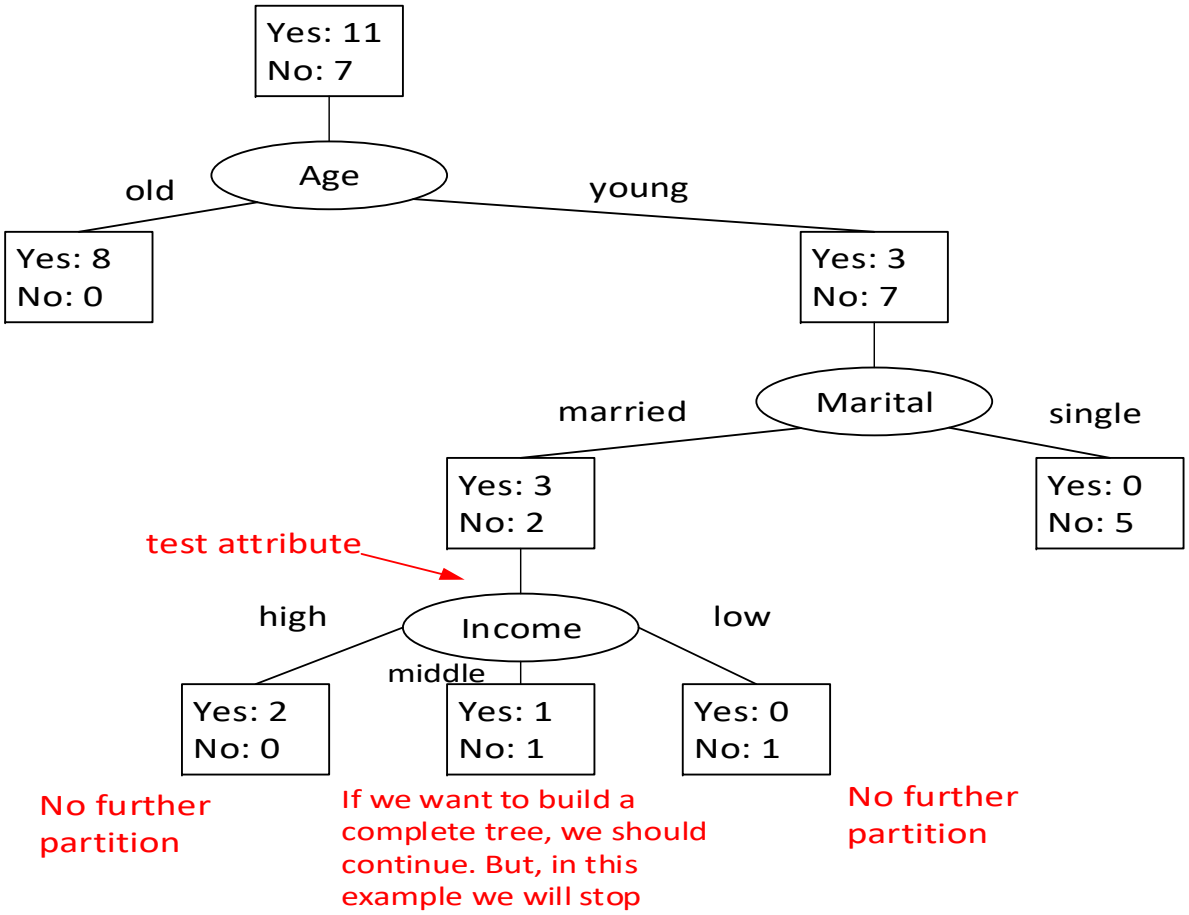
Algorithm for Decision Tree Induction

- The same process is repeated (recursively) on the right child node. The test attribute is *Marital*.



Algorithm for Decision Tree Induction

- The same process is repeated (recursively) on the left child node of *Marital*. The test attribute is *Income*.

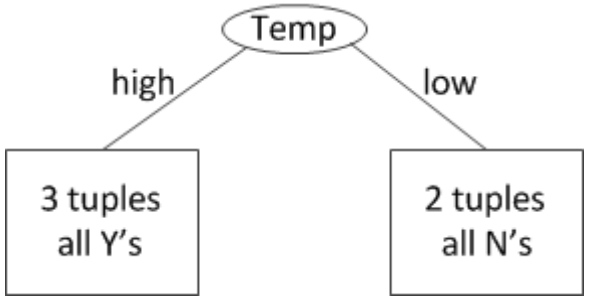


Attribute Selection Measure:

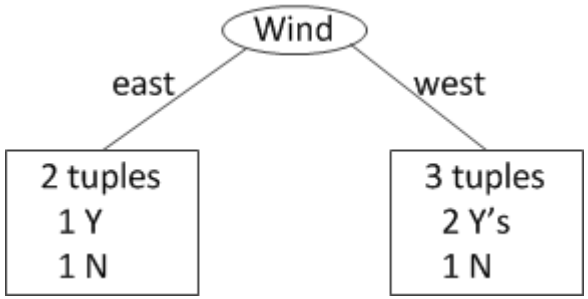
- Consider the following dataset:

Temp	Wind	Class
high	east	Y
low	west	N
low	east	N
high	west	Y
high	west	Y

Split by *Temp*



Split by *Wind*



- Which split is better?

Attribute Selection Measure:

- Test attribute is selected based on “purity measure.”
- Purity measures: information gain, gain ratio, Gini index, . . .

- Information gain is based on *info*.
- *Info* represents how pure a dataset is with regard to class labels.
- Suppose a dataset has 10 tuples with two class values, Y and N.
- If the dataset has all Y’s or all N’s, then the dataset is purest and the value of *info* is 0.
- If the dataset has 5 Y’s and 5 N’s, this is the extreme impure case, and the value of *info* is 1.

Attribute Selection Measure: Info Examples

- Notation: $I(x_1, x_2, \ldots, x_m)$, where

  m is the number of distinct class values,

  x_i is the number of tuples belonging to the i-th class

  $|D| = x_1 + x_2 + \ldots + x_m$

- Computation of *info*:

$$I(x_1, x_2, \ldots, x_m) = -\sum_{i=1}^{m} p_i \log_2(p_i), \text{ where } p_i = \frac{x_i}{|D|}$$

- *Info* is also referred to as *entropy*

Attribute Selection Measure: Info Examples

- Computing log base-2

$$\log_2 x = \frac{\log_{10} x}{\log_{10} 2} = \frac{\log_{10} x}{0.301}$$

$$\log_2 0.6 = \frac{\log_{10} 0.6}{\log_{10} 2} = \frac{-0.2218}{0.301} = -0.7369$$

Attribute Selection Measure: Info Examples

- A dataset has 6 tuples with 5 Y's and 1 N:

$$I(5,1) = -\frac{5}{6}\log_2(\frac{5}{6}) - \frac{1}{6}\log_2(\frac{1}{6}) = 0.650$$

- A dataset has 10 tuples with all Y's

$$I(10,0) = -\frac{10}{10}\log_2(\frac{10}{10}) - \frac{0}{10}\log_2(\frac{0}{10}) = 0$$

- A dataset has 10 tuples with 5 Y's and 5 N's

$$I(5,5) = -\frac{5}{10}\log_2(\frac{5}{10}) - \frac{5}{10}\log_2(\frac{5}{10}) = 1$$

Attribute Selection Measure: Information Gain (ID3)

- Compute the information gain of each attribute
- Select the attribute with the highest information gain
- Informal description of *information gain*:
  - *Info*(*D*): The amount of information we need to classify a sample in *D*.

    Example: we need 0.9
  - $Info_A(D)$: After splitting *D* on an attribute *A*, the amount of information needed to classify a sample.

    Example: after splitting on A, now we need only 0.3
  - $Gain(A) = Info(D) - Info_A(D)$.

    Example: the gain is 0.9 – 0.3 = 0.6

Attribute Selection Measure: Information Gain (ID3)

- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i, estimated by $|C_{i,D}|/|D|$
- Expected information (entropy) needed to classify a tuple in D:
  - *m*: # classes

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

- Information needed to classify D, after using attribute A to split D into v partitions:
  - *v*: # distinct values of *A*

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

- Information gained by splitting by A

$$Gain(A) = Info(D) - Info_A(D)$$

Example

- Building a decision tree from the RV dataset, *D*

Income	Marital	Age	Housing	Class
high	married	old	own	yes
high	married	young	own	yes
high	married	old	rent	yes
middle	married	young	own	yes
high	married	old	rent	yes
high	married	young	rent	yes
middle	married	old	rent	yes
high	single	young	own	no
high	single	old	own	yes
high	single	young	rent	no
low	married	old	own	yes
low	single	old	rent	yes
low	married	old	own	yes
low	single	young	own	no
middle	married	young	own	no
low	single	young	rent	no
low	married	young	rent	no
low	single	young	rent	no

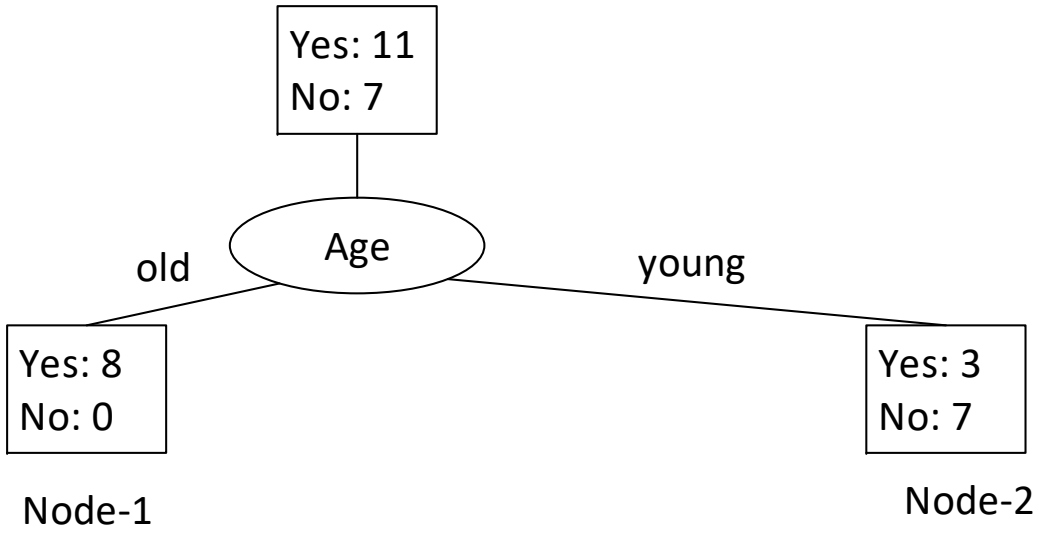
- There are 18 tuples in *D*
- 11 Yes's and 7 No's

$$Info(D) = I(11,7)$$
$$= -\frac{11}{18}\log_2(\frac{11}{18}) - \frac{7}{18}\log_2(\frac{7}{18})$$
$$= 0.964$$

Example

- Will illustrate the choice of the test attribute at the root level.
- Calculate information gains of all attributes and choose the one with the highest gain. Then, split the dataset by the test attribute.

- If we split *D* based on *Age*, two partitions are created.



Example

- Info of Node-1 is: $I(8,0) = 0$
- Info of Node-2 is: $I(3,7) = -\frac{3}{10}\log_2(\frac{3}{10}) - \frac{7}{10}\log_2(\frac{7}{10}) = 0.881$
- After splitting by *Age*, the amount of information needed to classify a tuple is computed as the weighted sum of the above two, where the weight is the fraction of tuples in a node.

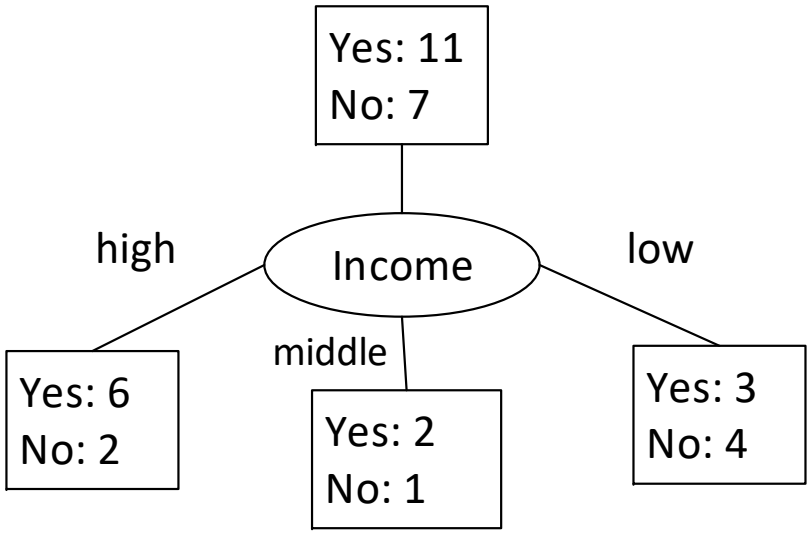
$$Info_{age}(D) = \frac{8}{18} I(8,0) + \frac{10}{18} I(3,7) = 0.489$$

- Information gain for *Age* is:

$$Gain(Age) = Info(D) - Info_{Age}(D) = 0.475$$

Example

- If we split by income, three children nodes are created:



$I(6, 2) = 0.811$
$I(2, 1) = 0.985$
$I(3, 4) = 0.918$

$$Info_{income}(D) = \frac{8}{18} I(6,2) + \frac{3}{18} I(2,1) + \frac{7}{18} I(3,4) = 0.903$$

$$Gain(income) = Info(D) - Info_{income}(D) = 0.061$$

Example

- We can compute *Gain*(*Marital*) and *Gain*(*Housing*) in the same way. Then, we have:

  Gain(*Age*) = 0.475

  Gain(*Income*) = 0.061

  Gain(*Marital*) = 0.210

  Gain(*Housing*) = 0.009

- We select *Age* as the test attribute because it has the highest information gain, and split *D* by *Age*.
- Repeat the same steps on the children nodes (as needed).

Gain Ratio for Attribute Selection (C4.5)

- Information gain measure is typically biased towards attributes with a large number of values.
- C4.5 (a successor of ID3) uses gain ratio to overcome the problem.

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

Here, attribute *A* has *v* distinct values, $|D_j|$ is the number of tuples with the *j*-th value.

- $GainRatio(A) = Gain(A)/SplitInfo_A(D)$
- The attribute with the maximum gain ratio is selected as the splitting attribute.

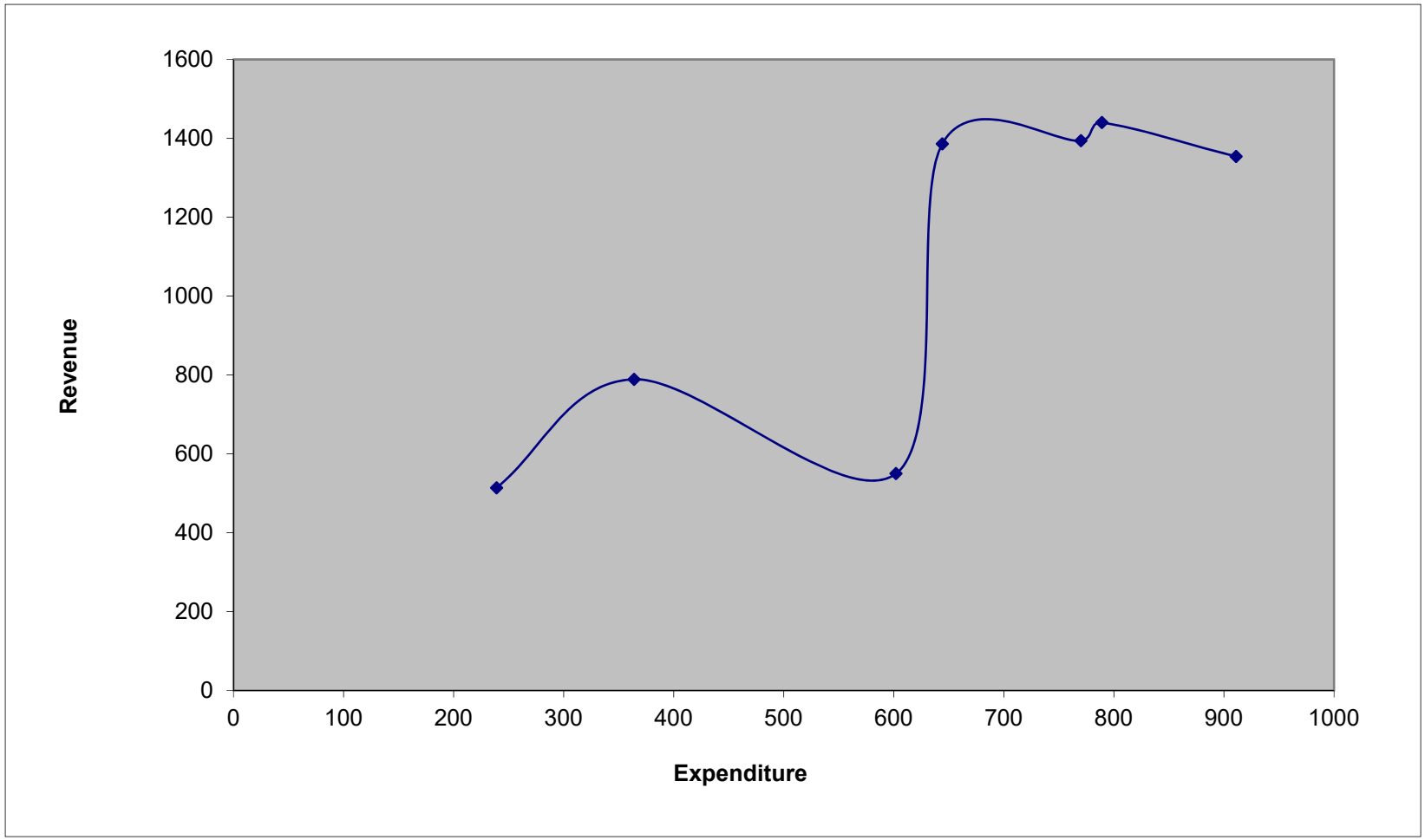
Other Attribute Selection Measures

- GINI index
- CHAID, C-SEP, G-statistic, MDL (Minimal Description Length) principle
- Multivariate splits (partition based on multiple variable combinations)
  - Example: CART
- Which attribute selection measure is the best?
  - Most give good results, none is significantly superior than others

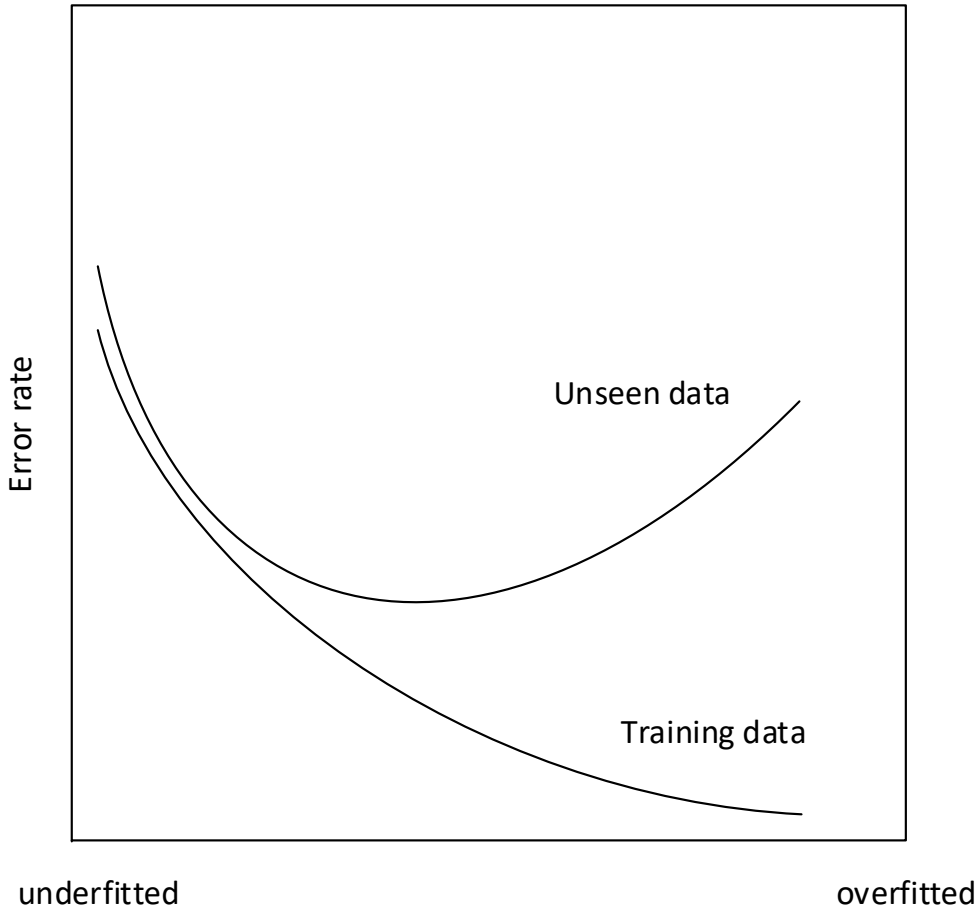
Overfitting

- Overfitting:
  - A model reflects every details of the training dataset, even an anomaly or noise.
  - A model becomes complex.
  - Accuracy on the training dataset is high.
  - Accuracy on the unseen dataset becomes low.
  - So, an overfitted model does not generalize well.

Overfitting



Overfitting



Overfitting and Decision Tree

- Overfitting of decision tree:
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Poor accuracy for unseen samples
- Two approaches to avoid overfitting
  - Prepruning: *Halt tree construction early* – do not split a node if this would result in the goodness measure falling below a threshold
    - Difficult to choose an appropriate threshold
  - Postpruning (more common): *Remove branches* from a "fully grown" tree – get a sequence of progressively pruned trees
    - Use a set of data different from the training data to decide which is the "best pruned tree"

Discretization Using Entropy

- A numeric variable can be discretized using entropy.
  - Supervised
  - Top-down method

References

- Han, J., Kamber, M., Pei, J., “Data mining: concepts and techniques,” 3rd Ed., Morgan Kaufmann, 2012
- http://www.cs.illinois.edu/~hanj/bk3/
- Ian H. Witten, E. Frank, and M.A. Hall, "Data Mining Practical Machine Learning Tools and Techniques," Third Ed., 2011, Morgan Kaufmann
- Shmueli, G., Bruce, P.C., Stephens, M.L., Patel, N.R., “Data mining for business analytics: concepts, techniques, and applications with JMP Pro,” Wiley, 2017.